DNA計算における部分グラフ同型問題の解法について (計算機科学基礎理論の新展開)

AUTHOR(S):

片山, 貴晴; 鵜飼, 亮介; 五所野尾, 一彦; 伊藤, 暢浩; 犬塚, 信博; 陳, 慰; 和田, 幸一

CITATION:

片山, 貴晴 ...[et al]. DNA計算における部分グラフ同型問題の解法について (計算機科学基礎理論の新展開). 数理解析研究所講究録 2003, 1325: 140-145

ISSUE DATE:

2003-05

URL:

http://hdl.handle.net/2433/43187

数理解析研究所講究録 1325 巻 2003 年 140-145

DNA計算における部分グラフ同型問題の解法について

片山 貴晴† 鵜飼 亮介† 五所野尾一彦† 伊藤 暢浩† 犬塚 信博†
陳 慰†† 和田 幸一†

† 名古屋工業大学電気情報工学科 〒466−8555 愛知県名古屋市昭和区御器所町
†† Tennesse State University, U.S.A. Department of Computer Science, Tennesse State University 3500 Jhon A Merrit Blvd, Nashville, TN 37205, USA
E-mail: †{taka, r_ukai, gosho, bobson}@phaser.elcom.nitech.ac.jp, {inuzuka, wada }@elcom.nitech.ac.jp, †† wchen@tnstate.ecu

1. はじめに

DNA 計算とは，DNA 分子の特徴を利用した分子計算の一種であり，1994 年に Adleman がハミルトン経路問題を実際に DNA 計算によって解いたことから始まった [1]. ハミルトン経路問題は NP 完全問題のひとつで，Adleman が NP 完全問題の計算可能性を示したことで，DNA 計算は世界中の研究者に注目されるようになった．ハミルトン経路問題を解くにあたり，Adleman は以下のような手法を用いた [1]. 最初に，大量の DNA を用いて，問題の解候補をすべて符号化する．そしてそれらを大量のプロセッサとして並列に動作させることにより，しらみつぶし探索を行うという手法である．

Adleman の実験では，解を符号化する DNA 鎖を探し出すために，抽出という操作を用いた．抽出とは，磁気を帯びたビーズなどを用いて，すべての解候補を符号化した DNA 鎖の中から，解の符号化となる DNA 鎖を「釣り上げる」ことにより，解候補を分離する操作である．しかし，この操作は非常にエラー率が高く，複数回抽出を行うと成功率が非常に低くなる．そのため，成功率を上げるためにより多くの時間と大量の DNA が必要となり，効率が悪い．

これを改善するため，Amos らによって除去という操作が提案された [2]. 除去操作は，解でない候補を符号化する DNA 鎖のみを制限酵素によって破壊し，解を符号化する DNA 鎖を残すことにより解候補を分離する操作である．抽出操作とは異なり，除去操作は化学的な手法を用いるので，エラー率は非常に低く，複数回の操作においても高い成功率を維持できる．

Amos らはこの除去操作を用いて，部分グラフ同型問題の解法を示した．部分グラフ同型問題とは，与えられた二つの無向グラフ G_1, G_2 において，G_2 が G_1 に含まれるかどうかを判定する問題である．Amos らの解法は，グラフ G_2 の全ての頂点に，グラフ G_1 の各頂点を対応付けた系列を解候補としている．これを頂点に基づく符号化による解法と呼ぶ．この解法において一度に必要な解候補の数は，G_1, G_2 それぞれの頂点数 n_1, n_2 に対し $n_1^{n_2}$ である．この解法は，解候補を頂点の系列によって符号化するので，グラフが疎である場合は，明らかに解でないものが大量に含まれている．そのため，解候補を符号化するために必要な DNA の量は，非現実的な量になる．

中本らは，解候補数を削減するために，辺に基づく符号化による解法を提案した [5]. この解法では，G_1, G_2 を有向化したものを G_1', G_2' とし，G_2' 上のあるオイラー閉路と，G_1' 上の経路を対応づけたものを辺の系列で表し，それを解候補とする．一度に必要な解候補の数は，G_1 の最大次数を d_m，G_2 の辺数を m_2 とすると，高々 $n_1 d_m^{2m_2-1}$ である．そのため，グラフ G_1, G_2 が疎である場合は，解候補数を非常に少なくできる．しかし，グラフが密の場合の解候補数は非常に多く，グラフによっては Amos らの手法より悪くなることも確認されている．

本稿では，使用する DNA の量を削減するために，二つの解法を提案する．一つは全域木を用いた解法，もう一つは分割統治法による解法である．

全域木を用いた解法は，中本らの辺に基づく符号化に加え，全域木を利用する解法である．中本らはグラフ G_2 全体のオイラー閉路の符号化を解候補に用いたが，提案する手法は全域木のみのオイラー閉路を用いることで，解候補の長さを短くし，解候補数を高々 $n_1 d_m^{2n_2-3}$ に減らすことが可能となる．

分割統治法による解法は，問題をより小さな問題に分割して，各小問題の解候補を生成し，それを統合するという手法を用いる解法である．すでに鵜飼，五所野尾らは，ハミルトン経路問題，グラフ 3 彩色問題に分割統治法を適用した手法を提案しており，両者とも同時に用いる解候補の数を大幅に減らせることを，計算機によるシミュレーションで確認している [3], [6].

また本稿では，頂点に基づく符号化による解法，辺に基づく符号化による解法，そして全域木を用いた解法のそれぞれに分割統治法を適用した解法を示す．この解法は，グラフを複数の小さな部分グラフに分割し，各部分グラフにおいて短い解候補を生成し，無駄な解候補を除去し，それらを統合してより長い解候補を生成していくというものである．Amos らや中本らのような，最初にグラフ全体を符号化する解候補をすべて生成する手法に比べ，この手法は，解候補の長さが短い内に無駄な解候補を除去するため，解候補の数を大きく減らすことができる．その結果，一度に必要な DNA の量を減らすことが可能となる．

更に本稿では，辺に基づく符号化による解法と，今回提案する全域木を用いた解法における必要 DNA 量を，計算機上のシミュレーションによって計測する．また，これらの三つの解法に分割統治法を適用した場合の，それぞれに必要な DNA の量も同様に計測する．この結果を表 1 に示す．計測方法は，グラフ G_1, G_2 の各頂点数を $n_1 = 16, n_2 = 8$ とし，(1) $m_1 = 16, m_2 = 9$, (2) $m_1 = 20, m_2 = 9$, (3) $m_1 = 16, m_2 = 13$ の三つの状態における各解法の必要 DNA 量を 50 回計測し，その平均を取った．結果は Amos らの解法に対する割合で表示した．

以上の結果，全域木を用いた解法では，辺に基づく符号化に

表 1 Amos の方法との必要 DNA 量における比較

状態	辺	全域木	頂点 (分割)	辺 (分割)	全域木 (分割)
(1)	49 %	0.42 %	0.20 %	0.00025 %	0.00018 %
(2)	780 %	3.5 %	0.11 %	0.0029 %	0.00081 %
(3)	730000 %	0.47 %	0.0026 %	0.00049 %	0.00011 %

よる解法と比較して，必要 DNA 量を少なくできることがわかった．これは G_2 の辺数が多いとき，特に顕著になる．しかし，全域木を用いた解法でも，G_1 が密の時には，必要 DNA 量は頂点に基づく符号化による解法よりも多くなることがある．

分割統治法を適用した場合では，いずれの解法も必要 DNA 量を減らすことができ，特にグラフの辺数が少ない場合に大きく減らせることがわかった．しかしながら，中本らの手法と，今回提案する全域木を用いた手法の場合，G_1 の辺数が多いと，解候補数は $n_1^{n_2}$ よりも多くなる場合が存在する．

2. DNA の構造，性質

DNA 分子はアデニン，グアニン，シトシン，チミンの 4 種類のデオキシリボヌクレオチドからなり，それぞれ A，G，C，T と表記する．これらが共有結合によって一本の鎖のように結合したものを一本鎖 DNA と呼び，5'-ATCG-3' のように向きを示して表記する．また，この並びを配列と呼ぶ．A と T，C と G は対をなし，水素結合を起こす．ある一本鎖 DNA が，他の一本鎖 DNA と完全に対をなすとき，それらは互いに相補的であるといい，一本鎖 DNA α に対して相補的な一本鎖を $\overline{\alpha}$ と表記する．互いに相補的な一本鎖同士が水素結合によって二重鎖になることをアニール (anneal) という．例えば，$\alpha = 5' - ATGA - 3'$ とすれば $\overline{\alpha} = 3' - TACT - 5'$ であり，両者は $\begin{smallmatrix} 5'-ATGA-3' \\ 3'-TACT-5' \end{smallmatrix}$ のようにアニールする．アニールの様子を図 1 に示す．

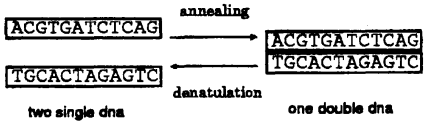

図 1　アニール操作

化学的に合成された比較的短い一本鎖をオリゴと呼び，二つのオリゴを連結するのに用いられるオリゴをリンカーと呼ぶ．例えば一本鎖 5'-ATCG-3' と 5'-GTTA-3' を連結する場合，前者の後半部分，後者の前半部分のそれぞれに相補的な配列を連結したオリゴ，つまり 3'-GCCA-5' がリンカーとなる．

3. 除去操作を用いる DNA 計算モデル

3.1 Amos の除去操作を用いたモデル

本稿では DNA 計算モデルとして除去操作による計算モデルを用いる [2]．除去操作による計算モデルでは，解を符号化していない DNA 鎖のみを切断し，DNA 鎖を長さについて分離することによって，試験管内に解を符号化したものだけを残すようにする．このような操作を用いる計算モデルとして，以下の六つの DNA 操作を定義する．ここでは，一つの集合は一つの試験管に相当し，その要素は一つの DNA 鎖に相当する．

- $union(\{U_1, U_2, \ldots, U_i\}, U)$
  複数の集合 $U_1, U_2, \ldots, U_i$ の合併部分を U にする．
- $copy(U, \{U_1, U_2, \ldots, U_i\})$
  集合 U を $U_1, U_2, \ldots, U_i$ に分ける．
- $select(U)$
  集合 U から無作為に一つの要素を選択する．
  もし U が空なら *empty* を返す．
- $amplify(U, n)$
  集合 U の複製を n 回作る操作であり，2^n 倍の複製が生成される．これは PCR 操作により実現される．
- $connect(U_1, U_2, U_3, L)$
  集合 U_1 と集合 U_2 の要素を，リンカー集合 L を用いて連結し，その集合を U_3 とする．どの要素に対しても十分な数が用意されているため，U_1 と U_2 の要素から生成されるようなすべての DNA 鎖の組み合わせが U_3 に生成されるものと仮定する．
- $remove(U, \{s_1, s_2, \ldots, s_i\})$
  集合 U に含まれる DNA 鎖のうち，配列 $s_1, s_2, \ldots, s_i$ のいずれかを自身の配列に含んでいるものを除去する．

copy 操作は，一つの集合 U を複数に分ける操作である．U にはすべての要素について十分な量の DNA 鎖が含まれており，分けられた試験管には U に存在する全ての要素が必ず含まれていると仮定する．

remove 操作は以下の二つの操作で実現される．

- $mark(U, s)$
  集合 U 内の DNA 鎖のうち，DNA 配列の一部に配列 s が含まれているもの全てにマークをつける．
- $destroy(U)$
  集合 U からマークのついた DNA 鎖全てを除去する．

本稿では，*mark* は次のようにおこなわれるものとする．まず，一本鎖 DNA の部分配列 s に相補的なオリゴを大量に用意し，それを集合 U に加える．すると，部分配列 s を含む DNA 鎖とオリゴがアニールし，一本鎖の一部に二重鎖が形成される．

次に，*destroy* は DNA 鎖の二重鎖部分のみに反応し切断する制限酵素 *Sau3AI* を加えることで実現する．マークのつけられた DNA 鎖を切断した後，ゲル電気泳動という技術によって，DNA 鎖を長さによって分離し，短い DNA 鎖を取り除き，必要な長さの DNA 鎖だけを取り出す．この様子を図 2 に示す．

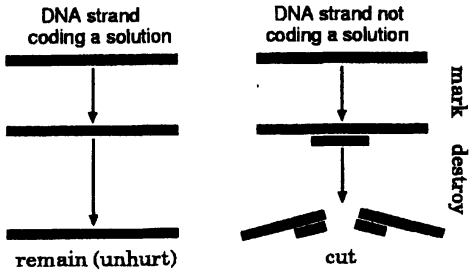

図 2　除去操作

本稿では，二つの試験管を混ぜ合わせる，すなわち二つの集合を一つの集合にまとめる操作を定数時間でおこなえるものとし，各操作の実行時間を以下のように仮定する．

$union(\{U_1, U_2, \ldots, U_i\}, U)$ 操作は，二つの集合を一つに，並列にまとめることを，集合が一つになるまで再帰的におこなうため，実行時間は $O(\log i)$ である．

$copy(U, \{U_1, U_2, \ldots, U_i\})$ 操作は，*union* 操作と逆のことを行なうため，実行時間は同じく $O(\log i)$ である．

select 操作は，集合から無作為に要素を一個選択する操作なので，$O(1)$ 時間で実行可能である．

$amplify(U, n)$ 操作の実行時間は PCR 操作の回数に比例するため，$O(n)$ 時間である．本稿では，集合を x 倍にする場合，$C \cdot x$ 倍 (C:定数) するようにしている．これは冗長的に複製を用意することで，*copy* 操作と *connect* 操作において起こりうる解候補生成のもれをなくすためである．

$connect(U_1, U_2, U_3, L)$ 操作は試験管 U_1，U_2，L を混ぜ合わせることで実行するため $O(1)$ 時間である．

$remove(U, \{s_1, s_2, \ldots, s_i\})$ 操作は，$s_1, s_2, \ldots, s_i$ に相補的な DNA 鎖と制限酵素を *union* 操作と同様にして一つの試験管にまとめ，それと試験管 U を混ぜ合わせることによっておこなうため，実行時間は $O(\log i)$ 時間である．

4. 部分グラフ同型問題の解法

部分グラフ同型問題とは，無向グラフ $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ $(|V_2| \leq |V_1|)$ において，G_1 の部分グラフの集合内に，G_2 と同型なグラフが存在するかどうかを判定する問題である．

同型なグラフが存在する条件は，$V_2 = \{v_1, v_2, \ldots, v_{n_2}\}$ であるとし，$V \subseteqq V_1, E \subseteqq E_1$ であるような G_1 の部分グラフ $G = (E, V)$ に対して，$(v_i, v_j) \in E_2 \Leftrightarrow (f(v_i), f(v_j)) \in E$ となる1対1の写像 $f : E_2 \to E$ が存在することである．本章では，DNA 計算による部分グラフ同型問題の解法を三つ示す．

4.1 頂点に基づく符号化による解法

Amos らが提案した，頂点に基づく符号化による解法[2]を示す．与えられた二つの無向グラフをそれぞれ $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ とし，$V_1 = \{u_1, u_2, \ldots, u_{n_1}\}, V_2 = \{v_1, v_2, \ldots, u_{n_2}\}$ とする．ただし，$n_2 \leqq n_1$ である．初期解候補は，$\hat{v}_1\hat{u}_{i_1}\hat{v}_2\hat{u}_{i_2}\ldots\hat{v}_{n_2}\hat{u}_{i_{n_2}}$ という形をしている．また，G_2 の頂点は固定されている．ここで，$1 \leqq i \leqq n_1$，$1 \leqq j \leqq n_2$ において $\hat{u}_i$，$\hat{v}_j$ はそれぞれ頂点 u_i, v_j の符号化を表し，$i_j \in \{1, 2, \ldots, n_1\}$ である．すなわち，$\hat{v}_j\hat{u}_i$ によって，グラフ G_2 の頂点 v_j とグラフ G_1 の頂点 u_i が対応していることを表す．

初期解候補は次に挙げるアルゴリズム Generating initial candidates によって生成することができる．ただし，試験管 $U_k(1 \leqq k \leqq n_1)$ には $\hat{v}_k\hat{u}_{i_k}$ で表される n_1 種類のオリゴが含まれており，試験管 $L_k(2 \leqq k \leqq n_1)$ には $\overline{\hat{u}_{i_{k-1}}\hat{v}_k}$ で表されるリンカーが含まれているものとする．

アルゴリズム Generating initial canditates

```
for k = 2 to n2 do begin
  amplify(U1,log (C·|Uk|))
  amplify(Uk,log (C·|U1|))
  amplify(Lk,log (C·|U1|·|Uk|))
  connect(U1, Uk, U1, Lk)
end
```

この生成過程は，G_1 の n_1 個の頂点から，重複を許して n_2 個を選ぶことに相当するので．初期解候補の数は $n_1{}^{n_2}$ である．生成される DNA 鎖の構造を図3に示す．

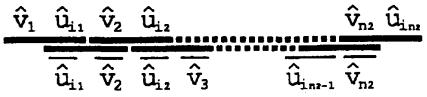

図3 頂点に基づく符号化における初期解候補

次に，得られた初期解候補を試験管 U に入れ，U に対して以下のような操作をする．

（1） 試験管 U に対して，G_1 の頂点と G_2 の頂点の対応が1対1になっていない DNA 鎖を除去する．

（2） （1）で残るもののうち，$(v_j, v_l) \in E_2$ であり $(u_{i_j}, u_{i_l}) \notin E_1$ であるような DNA 鎖を除去する．

（3） DNA 鎖の有無をチェックする．DNA 鎖が残っていれば，グラフ G_2 は G_1 に含まれる．

これらの操作は以下のアルゴリズム Isomorphism vertex で実現できる．ただし，C は定数である．

アルゴリズム Isomorphism vertex

```
for j = 1 to n2 − 1 do begin
  amplify(U, lg C·n1)
  copy(U, {U1, U2, ..., Un1})
  for i = 1, 2, ..., n1
    in parallel do begin
      remove(Ui, {v̂j û | û ≠ ûi})
      remove(Ui, ⋃k>j {v̂k ûi})
      remove(Ui, ⋃j<l≦n2 {v̂l û' | (vj, vl) ∈ E2 ∩ (ui, u') ∉ E1})
    end
  union({U1, U2, ..., Un1}, U)
end
select(U)
```

for in parallel do begin ～ *end* 間において各試験管を並列に処理できることから，このアルゴリズムの計算時間は $O(n_2 \log n_1)$ 時間と求められる．

4.2 辺に基づく符号化による解法

中本らの提案した，辺に基づく符号化による解法[5]を示す．まず，与えられた無向グラフ $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ に対して，それぞれ有向化したグラフを $G_1' = (V_1, E_1'), G_2' = (V_2, E_2')$ とする．ただし，G_2 は連結であると仮定して一般性を失わない[5]．次に，長さ $2m_2$ の G_2' 上のある一つのオイラー閉路を考え，その辺の列を生成する．そして，G_2' の各頂点に G_1' のすべての頂点を対応づけしたものを初期解候補とする．

初期解候補は以下の形式の DNA 鎖の集合である．

$$\hat{u}_{i_1}\hat{v}_{j_1}\hat{v}_{j_2}\hat{u}_{i_1}\ \hat{u}_{i_2}\hat{v}_{j_2}\hat{v}_{j_3}\hat{u}_{i_2}\ \cdots\hat{u}_{i_{2m_2}}\hat{v}_{j_{2m_2}}\hat{v}_{j_{2m_2+1}}\hat{u}_{i_{2m_2}}$$

ここで，$i_k \in \{1, 2, \ldots, n_1\}$，$j_k \in \{1, 2, \ldots, n_2\}$，$1 \leqq k \leqq 2m_2$ であり，$\langle v_{j_1}, v_{j_2}, \ldots, v_{j_{2m_2}}, v_{j_{2m_2+1}} = v_{j_1}\rangle$ は G_2' 上の固定した一つのオイラー閉路を表し，$\langle u_{i_1}, u_{i_2}, \ldots, u_{i_{2m_2-1}}, u_{i_{2m_2}}\rangle$ は G_1' 上の長さ $2m_2 - 1$ のパスを表す．これにより，G_2' の全ての頂点と辺を G_1' の辺と対応させることができる．

初期解候補の生成は，頂点に基づく符号化による解法と同様のアルゴリズムを用いておこなうことができる．ただし，用意する試験管の数は $2m_2$ 本であり，for ループの回数もそれに従って $2m_2 - 1$ 回になる．また，試験管 $U_k(1 \leqq k \leqq 2m_2)$ には，$\hat{u}_{i_k}\hat{v}_{j_k}\hat{v}_{j_{k+1}}\hat{u}_{i_k}$ で表される n_1 種類のオリゴがそれぞれ含まれているものとし，試験管 $L_k(2 \leqq k \leqq 2m_2)$ には，$\overline{\hat{v}_{j_k}\hat{u}_{i_{k-1}}\hat{u}_{i_k}\hat{v}_{j_k}}$ で表されるリンカーがそれぞれ含まれているものとする．生成アルゴリズムによって生成される初期解候補の形を図4に示す． 初期解候補数は G_1' 上の長さ $2m_2 - 1$ の経路の数に一致

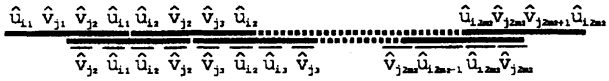

図4 中本らの解法における初期解候補

し，G_1' の最大出次数を d_m とすれば高々 $n_1 d_m^{2m_2-1}$ である．

初期解候補の入った試験管を U として，以下の操作を行う．

（1） 初期解候補において，配列 $u_{i_k}v_{j_k}(1 \leqq k \leqq 2m_2)$ は，G_1' の頂点 u_{i_k} と G_2' の頂点 v_{j_k} が対応していることを表している．ここで，G_1' の頂点と G_2' の頂点が1対1で対応していない DNA 鎖を U から除去する．

（2） DNA 鎖の有無をチェックする．

以上の操作は，次のアルゴリズム Isomorphism edge によって実行される．ただし，C は定数である．

アルゴリズム Isomorphism edge

```
for j = 1 to n2 do begin
  amplify(U, lg C·n1)
  copy(U, {U1, U2, ..., Un1})
  for i = 1, 2, ..., n1
    in parallel do begin
      remove(Ui, {ûi v̂ | v̂ ≠ v̂j})
      remove(Ui, {û v̂j | û ≠ ûi})
    end
  union({U1, U2, ..., Un1}, U)
end
select(U)
```

このアルゴリズムでは，6行目の *remove* 操作により，u_i と v_j を一対一に対応させている．すなわち，$u_iv(v \neq v_j)$ を除去することで，G_1' の頂点 u_i を G_2' 上の唯一の頂点 v_j と対応させ，$uv_j(u \neq u_i)$ を除去することで，G_2' の頂点 v_j が唯一対応するのは G_1' 上の頂点 u_i であるとしている．また，計算時間は頂点に基づく符号化と同様で $O(n_2 \log n_1)$ 時間となる．

4.3 全域木を用いた解法

本稿で提案する解法として，全域木を用いた解法を示す．

提案手法は，中本らの解法のように G_2' 全体のオイラー閉路を用いるのではなく，G_2 のある一つの全域木 T_2 を考え，T_2 を有向化したものを T_2' とし，そのオイラー閉路のひとつを用いるという手法である．ここで，m_2' は T_2 の辺数を表すとすると，オイラー閉路の長さは $2m_2' = 2n_2 - 2$ となる．

初期解候補は，中本らの解法とまったく同様である，

$$\hat{u}_{i_1}\hat{v}_{j_1}\hat{v}_{j_2}\hat{u}_{i_1}\ \hat{u}_{i_2}\hat{v}_{j_2}\hat{v}_{j_3}\hat{u}_{i_2}\ \cdots\hat{u}_{i_{2m_2'}}\hat{v}_{j_{2m_2'}}\hat{v}_{j_{2m_2'+1}}\hat{u}_{i_{2m_2'}}$$

という形式の DNA 鎖の集合を用意する．また，解候補の生成も中本らの解法と同様のアルゴリズムを用いておこなうことができる．ただし，用意する試験管の本数は $2m_2'$ 本であり，それに伴ってループ回数は $2m_2' - 1$ 回となる．また $m_2' = n_2 - 1$ なので，初期解候補数は高々 $n_1 d_m^{2n_2-3}$ である．

この手法におけるオイラー閉路の長さは G_2 の辺数ではなく頂点数に比例するため，解候補を符号化する DNA 鎖の長さは中本らの解法に比べて短くなる．しかし，この解候補は中本らの解法とは異なり，G_2' の辺を全て網羅しているわけではないので，アルゴリズムを若干変更する必要がある．

初期解候補の集合を U とし，以下の操作をおこなう．

（1） G_1' の頂点と G_2' の頂点が 1 対 1 で対応していない DNA 鎖を除去する．

（2） (1) で残るもののうち，$(v_j, v_k) \in E_2$ であり $(u_i, u_l) \notin E_1$ であるような DNA 鎖を除去する．ただし頂点 v_j は頂点 u_i と 1 対 1 対応し，頂点 v_k は頂点 u_l と対応しているものとする．

（3） DNA 鎖の有無をチェックする．

これらの操作は 4.2 節で示した除去アルゴリズムの内側の for ループの最後に，以下の操作を追加することで実行可能である．

$$remove(U_i, \bigcup_{j<k\leqq n_2}\{\hat{u}'\hat{v}_k \mid (v_j, v_k)\in E_2 \cap (u_i, u') \notin E_1\})$$

このアルゴリズムの計算時間は，$O(n_2 \log n_1)$ である．

5. 分割統治法の適用

本稿では，4 章で示した三つの解法それぞれに，分割統治法を適用した方法を示す．この解法は，これまでのように最初に全ての解候補を用意するのでなく，問題をいくつかの小問題に分割して解候補を生成し，随時除去アルゴリズムを適用し，残った解候補を統合してより大きな部分問題の解候補を生成する，といった手法を用いる．この手法によって，一度に必要な解候補の数を削減する．なお，分割の対象を，Amos らの解法はグラフ G_2，中本らの解法はグラフ G_2 のオイラー閉路，提案手法はグラフ G_2 の全域木 T_2 のオイラー閉路とする．

5.1 頂点に基づく符号化における分割統治法

Amos らの示した頂点に基づく符号化による解法に，分割統治法を適用した解法を示す．まず，試験管 $U_k(1 \leqq k \leqq n_2)$，$L_k(2 \leqq k \leqq n_2)$ は，4.1 節の頂点に基づく符号化において初期解候補を生成する際に用いたものと同じものであるとする．そして，分割統治アルゴリズム isomorphism Divide And Conquer(1, 解候補のサイズ) を実行し，*select* 操作をおこなうことで，G_2 が G_1 に含まれるかどうかを判定することができる．なお，解候補のサイズは，解候補を符号化する DNA 鎖に含まれる，$\hat{v}_k\hat{u}_{i_k}$ で表されるオリゴの数によって表すものとする．

分割統治アルゴリズム

Isomorphism Divide And Conquer$(k, size)$
  if $(size \geqq 2)$ **do begin**
    in parallel do begin
      U_k = **Isomorphism Divide And Conquer** $(k, size/2)$
      $U_{k+size/2}$ = **Isomorphism Divide And Conquer** $(k + size/2, size/2)$
    end
    in parallel do begin
      $amplify(U_k, \lg C \mid U_{k+size/2}\mid)$
      $amplify(U_{k+size/2}, \lg C \mid U_k\mid)$
    end
    $amplify(L_{k+size/2}, \lg(C \mid U_k \mid \cdot \mid U_{k+size/2}\mid))$
    $connect(U_k, U_{k+size/2}, U_k, L_{k+size/2})$
    解候補除去アルゴリズムによって，解候補を除去する．
  end
  return U_k

これは，左端の配列が $\hat{v}_k\hat{u}_{i_k}(1 \leqq k \leqq n_1, i_k \in \{1, 2, \ldots, n_1\})$ で表され，サイズが $size$ である解候補の集合を生成し，それを試験管 U_k に入れて返すアルゴリズムである．

アルゴリズムの流れは以下のようになる．

（1） 与えられた問題のサイズが 2 以上の場合

- 問題を半分に分割し，それぞれの部分問題における解を再帰的に得て，試験管 $U_k, U_{k+size/2}$ に入れる．
- 得られた解と，それらを連結するリンカーを増幅し，新たな解候補を生成し，U_k に入れる．
- U_k から解でない候補を除去する．

（2） そのサイズの問題における解として，U_k を返す．

次に 12 行目の，Amos らの解法に基づいた解候補除去アルゴリズム remove vertex を示す．これは，グラフ G_2 上の頂点 $v_x, v_{x+1}, \ldots, v_{x+size-1}$ からなる部分グラフに対応する解候補のうち，必要のないものを除去するアルゴリズムである．

アルゴリズム remove vertex$(U, x, size)$
  for $j = x$ to $(x + size - 1) - 1$ **do begin**
    $amplify(U, \lg C \cdot n_1)$
    $copy(U, \{U_1', U_2', \ldots, U_{n_1}'\})$
    for $i = 1, 2, \ldots, n_1$
      in parallel do begin
        $remove(U_i', \{\hat{v}_j\hat{u} \mid \hat{u} \neq \hat{u}_i\})$
        $remove(U_i', \bigcup_{k>j}\{\hat{v}_k\hat{u}_i\})$
        $remove(U_i', \bigcup_{j<l\leqq n_2}\{\hat{v}_l\hat{u}' \mid (v_j, v_l)\in E_2 \cap (u_i, u') \notin E_1\})$
      end
    $union(\{U_1', U_2', \ldots, U_{n_1}'\}, U)$
  end
  return U

分割統治アルゴリズムによって，問題はいったん n_2 個のサイズが 1 の部分問題に分割される．そして，各部分問題における解を並列に得，それらの二つずつを用いて新たな解候補を生成することを繰り返し行なう．この様子を図 5 に示す．

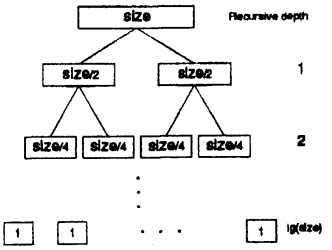

図 5 分割の図

この処理を，サイズが n_2 になるまで繰り返すと，最終的に Amos らの解法の初期解候補と同じ形式の解候補が生成できる．ただし，各ステップにおいて不要な解候補を随時除去しているため，解候補数は Amos らの解法のそれよりも少なくなる．

分割統治アルゴリズム Isomorphism Divide And Conquer(k,size) の計算時間は以下の漸化式を計算することで求

めることができる.

$$T(size) = T(\frac{size}{2}) + (\text{増幅,除去操作の計算時間}) \quad (1)$$

connect 操作は他の操作に比べて計算時間が非常に短いため，ここでは考えないものとする．最初の二つの *amplify* 操作は，互いの試験管の要素分だけ増幅する．解候補の *amplify* 操作は三つ目のものがもっとも増幅回数が多いため，この計算時間を求める．ここでリンカーを $C \cdot {n_1}^{size}$ 倍に増幅するため，*amlify* 操作の計算時間は $O(size \cdot \log n_1)$ である．

除去アルゴリズム remove vertex は Amos らの解法とほぼ同じものを利用しており，外側の *for* ループの回数を $size-1$ に変更しただけであるから，計算時間は $O(size \cdot \log n_1)$ である．

式 (1) より，分割統治アルゴリズム全体の計算時間は $T(n_2) = O(n_2 \log n_1)$ 時間である．

5.2 辺に基づく符号化における分割統治法

中本らの示した辺に基づく符号化による解法に，分割統治法を適用した解法を示す．

まず，最初に用意する試験管 $U_k(1 \leqq k \leqq 2m_2)$，$L_k(2 \leqq k \leqq 2m_2)$ は，4.2 節の辺に基づく符号化による解法において初期解候補を生成する際に用いたものと同様のものを用意する．

分割統治アルゴリズムは，Amos らの解法に適用したものと同じものを使うことができる．ただし，解候補のサイズ (size) は Amos らの解法の場合とは異なり，解候補を符号化する DNA 鎖に含まれる，$\hat{u}_{i_k}\hat{v}_{i_k}\hat{v}_{i_k+1}\hat{u}_{i_k}$ で表されるオリゴの数によって表す．また除去アルゴリズムは，4.2 節で示した解候補除去アルゴリズムにおいて，*select*(U) を **return** U に変更したものを使用する．それによって，G_2 のオイラー閉路を構成する経路 $\langle u_{i_x}, u_{i_{x+1}}, \cdots, u_{i_{x+size-1}} \rangle$ に対応した解候補から，不要なものを除去することができる．

次に，最悪の場合におけるアルゴリズム全体の計算時間を，式 (1) を利用して求める．ここで，最悪の場合とは，グラフ G_1, G_2 がともに完全グラフであると仮定する．このため，$m_1 = n_1(n_1-1)/2, m_2 = n_2(n_2-1)/2, d_m = n_1 - 1$ である．

分割統治アルゴリズムの 3 番目の *amplify* 操作における増幅回数は，各試験管の解候補数が高々 $n_1 d_m^{size-1}$ であるから，$2\lg(Cn_1 d_m^{size-1})$ と求めることができる．仮定より，$d_m = n_1 - 1$ であるから，これは $2\lg\{Cn_1(n_1-1)^{size-1}\}$ と書くことができる．これを計算すると，*amplify* 操作の計算時間は $O(size \cdot \log n_1)$ と求めることができる．

解候補除去アルゴリズムは分割統治法を適用する前と後で変わっていないので，アルゴリズム remove edge の計算時間は 4.2 節で示したものと同じ $O(n_2 \log n_1)$ である．

式 (1) より，$T(2m_2) = T(n_2(n_2-1))$ を計算すると，この解法における計算時間は最悪の場合 $O({n_2}^2 \log n_1)$ である．

5.3 全域木を用いた解法における分割統治法

全域木を用いた解法に，分割統治法を適用する．まず，用意する試験管は解候補生成に使用した $2m_2'$ 本の物と同じ試験管である．また，分割統治アルゴリズムも同じであり，除去アルゴリズムは 4.3 節のアルゴリズムにおける *select*(U) を **return** U に変更したものを使用する．

最悪の場合におけるアルゴリズム全体の計算時間を，式 (1) を利用して計算する．ここで，最悪の場合とは，グラフ G_1, G_2 がともに完全グラフの時である．したがって，$m_1 = n_1(n_1-1)/2, d_m = n_1 - 1$ である．ただし，m_2' は G_2 の全域木 T_2 の辺数を表しているので，$m_2' = n_2 - 1$ である．

amplify 操作における解候補の増幅回数は，中本らの解法と同様に計算すると，$2\lg\{Cn_1(n_1-1)^{size}\}$ となる．よって，*amplify* 操作の計算時間は $O(size \cdot \log n_1)$ である．

除去アルゴリズムは 4.3 節で示した提案手法のアルゴリズムとほぼ同じであるため，計算時間は $O(n_2 \log n_1)$ である．

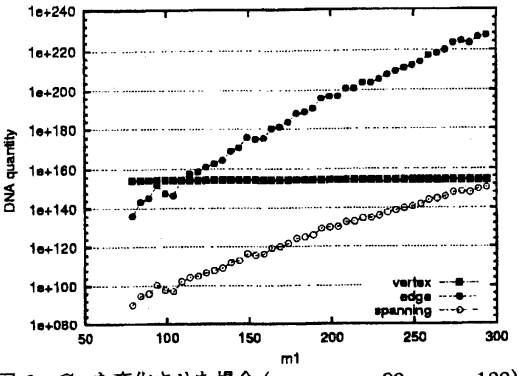

図 6 G_1 を変化させた場合 ($n_1 = n_2 = 80, m_2 = 120$)

$size = 2m_2' = 2n_2 - 2$ より，式 (1) から分割統治アルゴリズム全体の計算時間は $O(n_2 \log n_2 \log n_1)$ と求めることができる．

6. シミュレーション方法と結果

辺に基づく符号化による解法や分割統治法による解法では，解候補の数はグラフに依存するため直接計算することができない．そこで，実際に規模の小さな問題を解く場合に必要な DNA の量を，電子計算機によるシミュレーションを用いて計測する．

本論文では以下のシミュレーションをおこなった．

(1) 頂点に基づく符号化による解法，辺に基づく符号化による解法，全域木を用いた解法における必要 DNA 量の比較

(2) 分割統治法を用いた場合の必要 DNA 量の比較

(3) G_2 を木に限定したとき（この場合も NP-完全である）に分割統治法を用いた場合の必要 DNA 量の比較

なお，必要 DNA 量 = 同時に扱う DNA 量の最大量とし，DNA 量は (解候補数)×(解候補に含まれる頂点の数) とする．

(1)(2) では，G_1 と G_2 がともに連結であるとし，G_1 の辺数を変化させた場合と G_2 の辺数を変化させた場合についてシミュレーションをおこなう．

(3) では，G_1 は一般グラフ，G_2 は木として，G_1 の辺数を変化させてシミュレーションを行う．ここでは G_2 が木なので，辺に基づく符号化による解法と全域木を用いた解法は，解候補数が変わらない．よって，頂点に基づく符号化と辺に基づく符号化による解法による必要 DNA 量のみを示す．

(1) において，G_1 の辺数のみを変化させて計測した場合，図 6 のようになった．このことから，全域木を用いることによって，解候補を大幅に減らせることがわかる．

(2) において，G_1 の辺数のみを変化させて計測した場合，図 7,8 のようになった．図 7,8 から，いずれの解法も必要 DNA 量を大きく減らし、特に G_2 が疎の場合には，辺に基づく符号化は，必要 DNA 量を大きく減らすことができた．しかし，G_2 が密の場合にはそれほど減らすことができなかった．また，全域木を用いた解法では，どちらの場合も他の解法より少なかった．

次に G_2 の辺数のみを変化させた場合の計測結果を図 9,10 に示す．これらの図からも，G_1 が疎であれば辺に基づく符号化と全域木を用いた解法がより優れていることがわかる．

(3) における計測結果を図 11,12 に示す．グラフ G_2 の頂点数が G_1 に比べて少ない場合は，辺に基づく符号化の方がはるかに必要 DNA 量が多くなり，G_2 の頂点数が多い場合には両者とも同じ程度になるという結果になった．

7. まとめと今後の課題

本稿では，DNA 計算に必要な DNA の量を減らすために，文献 [5] で示された辺に基づく符号化による解法を改良した，全域木を用いる解法を用いた．この解法によって，辺に基づく符号化の弱点であったグラフが密の場合における必要 DNA 量をある程度改善することができた．そして，初期解候補の数を辺

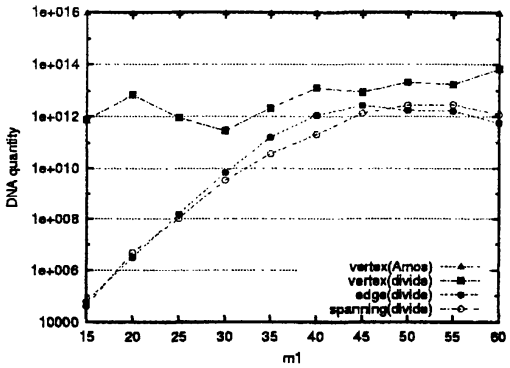

図 7 分割統治，G_1 を変化させた場合 $(n_1 = 16, n_2 = 12, m_2 = 12)$

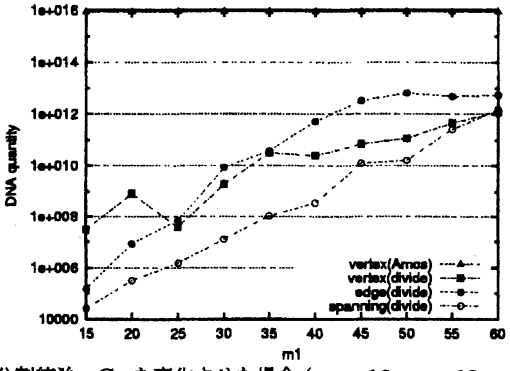

図 8 分割統治，G_1 を変化させた場合 $(n_1 = 16, n_2 = 12, m_2 = 24)$

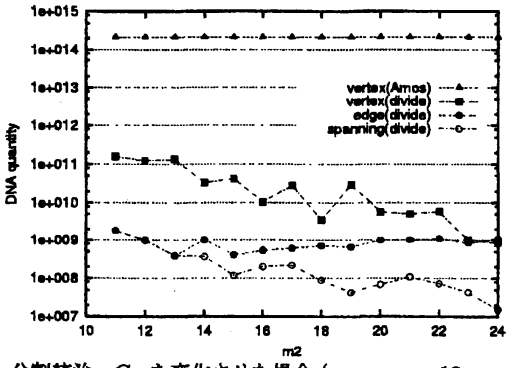

図 9 分割統治，G_2 を変化させた場合 $(n_1 = n_2 = 12, m_1 = 24)$

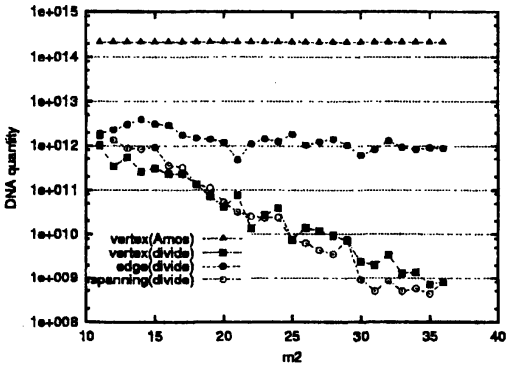

図 10 分割統治，G_2 を変化させた場合 $(n_1 = n_2 = 12, m_1 = 36)$

に基づく符号化以上に削減することができた．

また，それらの解法に分割統治法を適用することで，同時に必要な DNA の量を大きく減らすことができた．しかしほとんどの場合において，頂点に基づく符号化による解法が他の二つの解法よりも必要 DNA 量を多く減らしているという結果になった．特にグラフが密の場合，その傾向が如実に現れている．場合によっては，辺に基づく符号化による解法に分割統治法を用いた場合の解候補数が，頂点に基づく符号化で分割統治法を

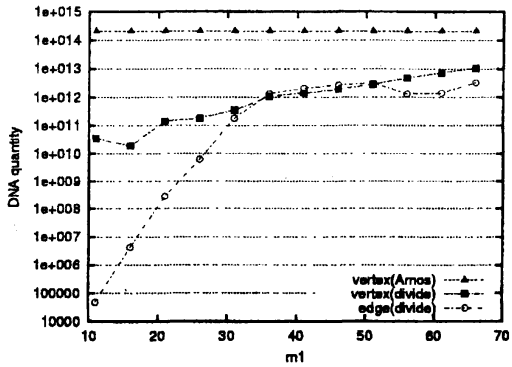

図 11 G_1 が一般グラフ，G_2 が木の場合に分割統治法を適用した場合 $(n_1 = 12, n_2 = 12, m_2 = 11)$

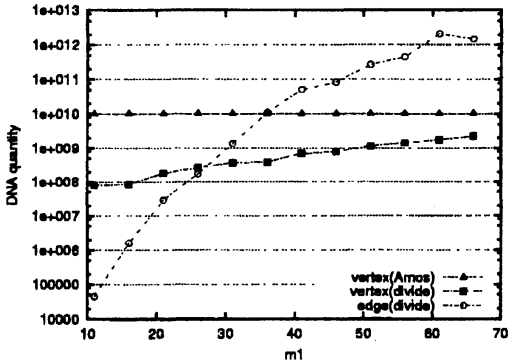

図 12 G_1 が一般グラフ，G_2 が木の場合に分割統治法を適用した場合 $(n_1 = 12, n_2 = 8, m_2 = 7)$

用いない場合よりも多くなってしまうということもある．

現在，頂点に基づく符号化による解法よりも常に解候補数を少なくすることのできる手法を示している [4]．また，分割統治法以外の解候補数削減手法の考案や，符号化方法の改良などの課題が残されている．

文　献

[1] Leonard Adleman:
"Computation of Solutions to Combinatorial Problems",
Science,266:1021-1024,November 1994.

[2] Martyn Amos, Alan Gibbons, and David Hodgson:
"Error-resistant Implementation of DNA Computations",
Research Report CS-RR-298, Department of Computer Science, University of Warwick, Coventry, UK, January 1996.

[3] 五所野尾 一彦，鵜飼 亮介，伊藤 暢浩，犬塚 信博，陳 慰，和田 幸一:
"DNA 計算におけるグラフ三彩色問題への分割統治法の適用について",
電子情報通信学会技術研究報告 COMP2002-35 信学技報 Vol.102 No.343:41-48.

[4] 片山 貴晴
"DNA 計算による部分グラフ同型問題の解法における使用 DNA 量の削減方法について",
平成 14 年度 名古屋工業大学卒業論文

[5] 中本 聡，五所野尾 一彦，陳 慰，和田 幸一:
"DNA 計算におけるグラフ問題の符号化について",
電子情報通信学会論文誌 (2002),Vol.J85-D-1 No.5,393-401.

[6] 鵜飼 亮介，五所野尾 一彦，伊藤 暢浩，陳 慰，和田 幸一:
"除去操作を用いた DNA 計算におけるハミルトン経路問題の解法について",
電子情報通信学会技術研究報告 COMP2002-13 信学技報 Vol.102 No.90:33-40.